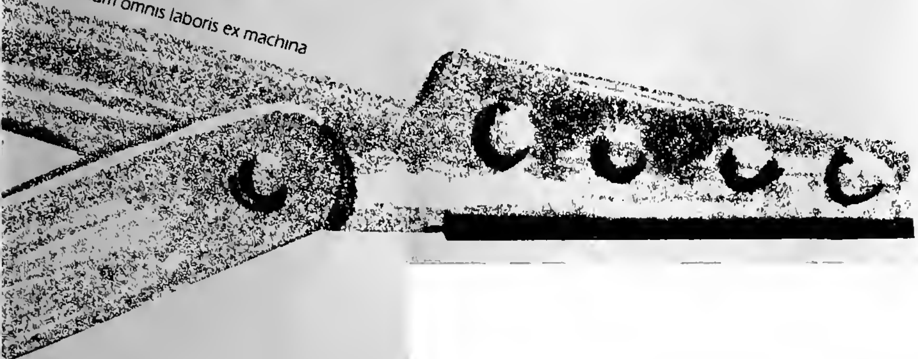
Robotics Research Technical Report

Generatorium omnis laboris ex machina

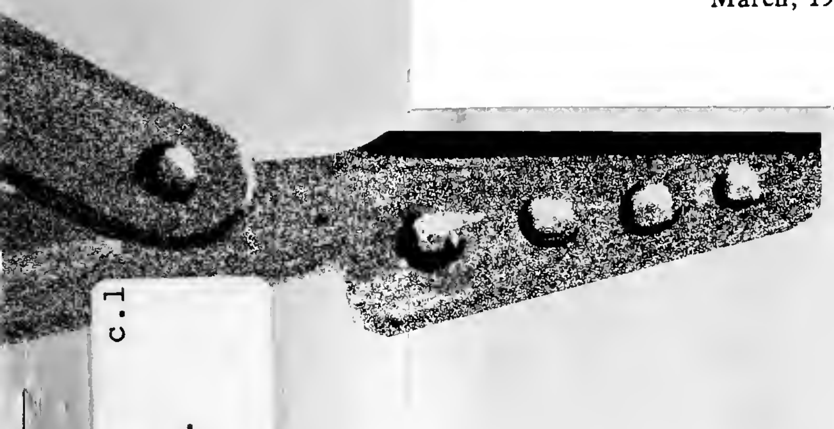


3-D Curve Matching

by

Eyal Kishon
Haim Wolfson

Technical Report No. 283
Robotics Report No. 103
March, 1987



New York University
(t Institute of Mathematical Sciences

Computer Science Division
251 Mercer Street New York, N.Y. 10012

3-D Curve Matching

by

Eyal Kishon
Haim Wolfson

Technical Report No. 283
Robotics Report No. 103
March, 1987

New York University
Dept. of Computer Science
Courant Institute of Mathematical Sciences
251 Mercer Street
New York, New York 10012

Work on this paper has been supported by Office of Naval Research Grant N00014-82-K-0381, National Science Foundation CER Grant DCR-83-20085, and by grants from the Digital Equipment Corporation and the IBM Corporation.

3-D Curve Matching

Eyal Kishon and Haim Wolfson[*]
Robotics Research Laboratory
Computer Science Department
Courant Institute of Mathematical Sciences
New York University

March 9, 1987

Abstract

An algorithm to find the longest common subcurve of two 3-D curves is presented. This algorithm is of average complexity O(n) where n is the number of the sample points on the two curves. Applications to part assembly and object recognition problems are discussed. Experimental results are included.

1 Introduction

1.1 Object Recognition

Object recognition is a major task in robotic vision. In a factory environment one is usually faced with the restricted problem of model based object recognition, since we expect the robot to see only a certain subset of factory tools and manufactured parts. Under this assumption we have to solve two major problems which are interrelated. The first is the *'model acquisition'*.

[*]Work on this paper was supported by Office of Naval Research Grant N00014-82-K-0381 and National Science Foundation Grant No. NSF-DCR-83-20085.

or *'data-base formation'* problem. The second is the actual *'object recognition'* problem, which will use a previously prepared data-base. The model descriptions used must be rich enough for recognition purposes; however, we would like them be terse to enable an efficient recognition process. An efficient object recognition algorithm is apt to rely on a favorable model description.

Existing object recognition systems (see [BJ85], [CD86]) use either 2-D or 3-D models. 3-D descriptions have the advantage of allowing a full description of a 3-D model from an unconstrained viewpoint. In an industrial application we can usually expect to be in a position to obtain this description, since the model objects can be preprocessed by taking pictures. Other sensory information may be available as well. Since a natural description of a 3-D object is by its bounding surfaces, much prior work concentrates on object recognition using surface information (see the surveys cited above). However, a reliable surface description requires a large amount of data. Handling this data. even using efficient algorithms, is likely to make recognition time consuming.

Use of surfaces for object description maybe motivated by an implicit concern for object reconstruction. In order to reconstruct a 3-D object, it is sufficient to know all its bounding surfaces. However, the object recognition problem is really more restricted. All we need, is to decide to which one of the models taken from a data-base which is usually not very big, it fits best. However, the object itself can be observed from an arbitrary viewpoint and may be partially occluded. For this reason we have to describe the objects in a way which can distinguish them given an unknown viewpoint and partial occlusion. We believe that the curve based techniques described below provide a rich enough description for these purposes, and also allow efficient

handling (almost real time). Various sorts of curves on 3-D objects can be used for this. These may be curves of occlusion (i.e. object boundaries), curves of intersection between neighbouring surfaces, lines of maximal local curvature, and artificially painted curves. Of course, there are surfaces on which there are almost no significant curves. In such cases surface information is essential.

1.2 Related Work

Our work exploits efficient curve matching algorithms which were developed for the 2-D recognition problem. In this simpler situation objects are uniquely defined by their boundary curves, hence the use of curves for object description is very natural. In order to recognize a partially occluded curve as belonging to a specific model in a 2-D situation, we need an algorithm which finds the best possible match between a curve and its proper subcurve. An algorithm due to Schwartz and Sharir (see [SS 85]), first implemented in the 2-D case, has proved to be very robust, as indicated by its successful application in visual assembly of a 200-piece jigsaw puzzle (see [KLSW 86]). This algorithm has been implemented for the 3-D case as well ([BSSS 86]), and also extended to the case in which several subcurves have to be matched simultaneously against the same curve ([K 86]). However, this algorithm requires knowledge of the exact starting point and endpoint of the observed subcurve. Such information is not always available in composite overlapping scenes of objects. When it is not, a more general curve matching algorithm is required, namely, we have to solve the following problem :

> *given two curves, find the longest matching subcurve which appears in both curves.*

Several algorithms that solve this problem for the 2-D case were proposed in [W 86]. The idea of that paper is to transform 2-D curves into strings of local, rotationally and translationally invariant shape signatures, apply efficient string matching techniques to discover starting points and end-points of long matching subcurves, and then match these subcurves using the Schwartz-Sharir curve matching algorithm. The approach to the 3-D curve matching problem described in this paper is quite similar. Again we transform the curves into sequences of local, rotationally and translationally invariant shape signatures. These signatures are actually k-tuples based on k signatures taken at different resolutions. Then we apply a geometric hashing technique to find candidate long matching subcurves. Finally, we choose just one of the several candidates by using the 3-D version of the Schwartz-Sharir algorithm, which also computes the transformation which has to be applied to this curve in order to align it with the other along their longest matching subpart.

The geometric hashing technique (so called, 'footprint' method) for curve matching that we generalize was first proposed in [KSSS 87] and later improved in [HW]. We use this later version of the technique. One of the major advantages of this technique is that it makes it possible to match an observed curve against a large data-base of other curves in average time linear in the number of sample points of the observed curve. Thus it enables us to tackle the more general recognition problem :

> *given a database of model curves, and an observed curve, find the model curve in the database with the longest matching subcurve with the observed curve.*

Another significant advantage of this technique is its straightforward parallelism.

1.3 Practical Implementation

At this stage in our work we have restricted our experiments to matching of curve pairs. This has other applications besides the object recognition problem. One such application is the problem of reassembly of broken objects. For example we have assembled the fractured 3-D object in Figure 1 by matching the boundary curves of the different pieces. Another possible application is to obtain a full description of a 3-D curve from a small number of overlapping partial views (curve reconstruction). Suppose, more specifically, that we are given two pictures (including depth information) of the same object from different viewpoints, so that the scenes overlap. We would like to merge the information obtained from both pictures into one consistent model. The obvious way to merge the curve information is by joining the appropriate curves along their longest matching subcurve. This technique will enable us to build a description of an object model, based on its curves, by taking pictures from a small number of viewpoints.

1.4 Organization

This paper is organized as follows. Section 2 describes the shape signatures we have experimented with. Section 3 describes our geometric hashing algorithm. Section 4 presents experimental results. In section 5 we suggest some directions for future research.

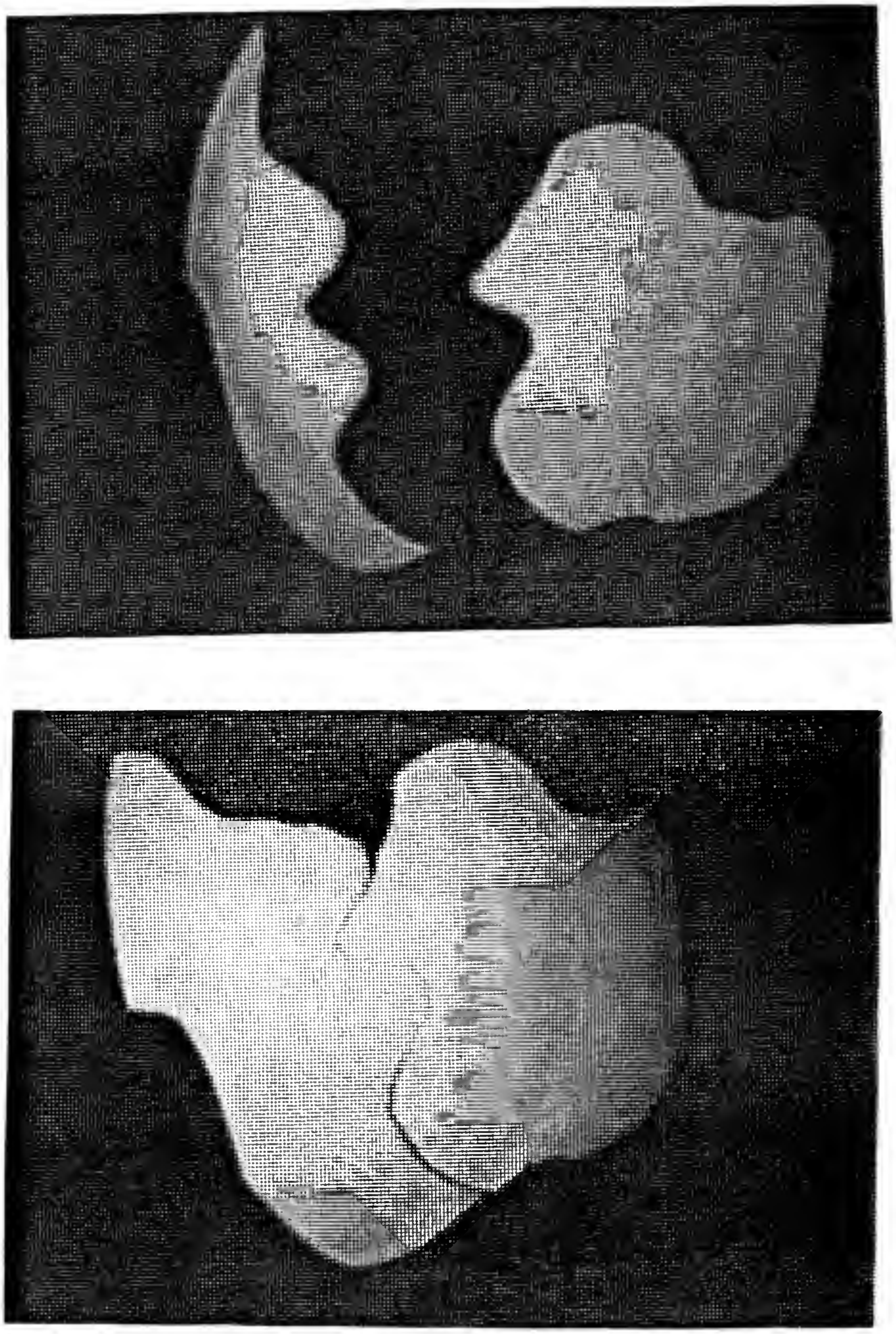


Figure 1: a) two pieces of a plastic ball. b) the two pieces assembled together.

2 Shape Signatures

2.1 Signature Properties

In this section we describe the curve signatures which are used in our matching algorithm. The signatures are used to compare relatively short segments of the curves to each other, hence we require them to be

i) local,

ii) translationally and rotationally invariant,

iii) stable, in the sense that small changes in the curve induce small effects (or no effect at all) on the associated signatures,

a further desirable, but less essential, property is :

iv) an approximation to an observed curve can be reconstructed from its signature sequence.

Although reconstruction is non essential for the recognition problem, we desire a weaker property which is important for successful recognition, namely, separation, to wit :

iv') essentially different local curve segments will create different signatures.

It is well known from Differential Geometry that smooth curves can be uniquely reconstructed within a rigid motion (i.e rotation and translation) using three geometric invariants, which are arc length s, curvature $\kappa(s)$ and torsion $\tau(s)$ as a function of s (see, for example, [D 76] or [S 69]). For ideal differentiable curves curvature and torsion are therefore suitable candidates

for curve signatures. However, our applications must deal with noisy polygonal representations of curves, making it impossible to compute curvature and torsion either accurately, or at every point of a curve. This introduces a problem of numerical stability. Curvature is essentially a second order derivative, and torsion involves even higher order derivatives. It is known that computations of (approximated) derivatives of noisy data are numerically unreliable. Hence we have decided not to use (an approximation to) torsion in our experimental signature.

Given this decision we have experimented with two signature schemes, which can best be explained using the following notation. We represent a point on a curve C by the vector $X(s) = (x(s), y(s), z(s))$, where s is the arclength. Since our curves are sampled, curves are represented by a sequence of sample points (X_i, $i = 1, ..., n$).

2.2 Curvature Based Signatures

The first signature we experimented with was based on curvature alone. The curvature of an arc $X(s)$ ($\alpha \leq s \leq \beta$) can be defined as the length of the curve traced out by the normalized tangent $\mathbf{t}(s) = dX/ds/\|dX/ds\|$ on the unit sphere (the so called Gaussian sphere) as s varies over the arc (see [S 69] p.54).

Thus, calculation of approximate local curvature can be accomplished as follows:

1. take equally spaced (by arclength) sample points ($X_i(s)$, $i = 1, ..., n$) on the curve;

2. at each sample point compute an approximate tangent vector at this point $T_i = X_{i+1} - X_{i-1}$, $i = 2, ..., n-1$;

3. normalize the tangents: $\mathbf{t_i} = T_i/||T_i||$; (This gives us points on the Gaussian sphere.)

4. compute the distance between successive normalized tangents : $\kappa_i = ||\mathbf{t_i} - \mathbf{t_{i-1}}||$, $i = 3, ..., n-1$; (This gives approximation of the geodesic distance between the normalized tangents on the Gaussian sphere .)

It is clear that this signature, although only an approximation to local curvature, satisfies properties (i)-(iii), emphasized above. To improve separation we define our signature to be as a k-tuple (in our experiments k=5) of local curvatures defined as just explained. In the actual computation we apply a multi-resolution approach, so that the relevant sample points for our local curvature measurements are taken at successively increasing distances, namely, the signature vector at point i is defined as

$\mathcal{S}_i = (\kappa_{i,1}, \kappa_{i,2}, ..., \kappa_{i,k})$,

where $\kappa_{i,j}$ $(j = 1, ..., k)$ is generated as explained above using the tangent approximation $T_{i,j} = X_{i+j} - X_{i-j}$; $i = k+1, ..., n-k$; $j = 1, ..., k$.

At each sample point on the curve we therefore get a k-dimensional vector (k-signature) representing the multi-resolution curvature measurements.

2.3 Tangent Magnitude Based Signatures

As noted above, a problem of numerical stability arises when derivatives of noisy data are computed. The previous paragraph describes a signature which is based on an approximation of a second order derivative. Obviously a signature based on a first order derivative should be more stable. This leads as to consider a second signature, based on the norm of the tangent

vector at a curve point, namely $\|X'(s)\|$. To approximate this signature at a sample point X_i we simply take the Euclidean norm of the difference vector $T_i = X_{i+1} - X_{i-1}$. Here again we compute k-signatures (in our experiment again k=5) using the multi-resolution approach analogous to the previously explained example. The thereby formed signature is obviously local, and translationally and rotationally invariant. Our experiments show this second signature to be more efficient than the curvature based signature, probably because of its greater computational robustness.

It is obvious that our signature sequences cannot describe a curve uniquely. However, as will be seen from the description of the matching algorithm in the next section, we only use these signatures in the first stage of the algorithm to filter out unsuitable candidates for matching. The reduced number of candidates that survive this filtering procedure are then matched by the robust 3-D matching algorithm due to Schwartz and Sharir, which takes into account all 3-D information about given sample points.

3 The Matching Algorithm

In this section we describe the matching algorithm applied in our experiments and the geometric hashing used. This technique was introduced for the 2-D case in [KSSS 87], and later improved in [HW]. The approach of [HW] is well tailored for partial curve matching and is used in our 3-D experiments. This method can be applied to the problem of matching one curve against another (*two-curve matching*), as well as to the problem of matching an observed curve against a large data-base of model curves. Al-

though the purpose of our experiment was two-curve matching, we describe the algorithm in general terms to emphasize that it is suitable for use in the general object recognition problem (see discussion in the Introduction).

The algorithm consists of two major steps. The first one is a preprocessing step which is applied to the data base of model curves. The complexity of this step is linear in the total number of sample points of the curves in the data base. This step is executed off-line before actual matching is attempted. The second step, matching proper, uses the data prepared by the first step and can be executed in time which, on the average, is linearly dependent on the number of sample points on the observed curve, thus achieving matching in time almost independent on the size and number of curves in the data-base.

3.1 Preprocessing

All the curves in the data-base are processed as follows. The curve is sampled and shape signature values are computed at each sample point, using the signature generation process described in the previous section. Note again that this produces k-tuples of signatures representing local, rotationally and translationally invariant characteristics of the curve (so called, *'footprints'*). For each such k-signature we record the *curve number and the sample point number* at which this signature was generated. This data is held in a hash-table, whos entries are k-dimensional vectors. (Of course, vector coordinates must be properly quantized to make the number of entries finite.) Successive signatures along the curve have a natural order defined by the way the curve is traced. Preprocessing time is linearly dependent on the total of sample points on the data-base curves. New curves added to the data-base can be processed independently without

recomputing the hash-table (except when we must re-hash).

3.2 Matching

In the matching stage an observed curve is sampled and k-signatures are computed at the sampling points. For each such signature we check the appropriate entry in the hash-table, and for every pair of (*model curve number, sample point number*), appearing there we add a vote for this model curve and the relative shift between the model curve and the observed curve. For example, if a signature, which was computed at the i'th sample point on the observed curve, appeared on model curves k_1 and k_2 at sample points j_1 and j_2 respectively, we add votes to model curve k_1 with relative shift $i - j_1$ and model curve k_2 with relative shift $i - j_2$. Obviously, long matching subcurves will cause a large number of signature coincidences between the appropriate curves. However, we look for *consistent coincidences*, which are singled out by identical relative shifts between signature sequences. (Because the signature coordinates are truncated we make the check described not only for the appropriate entry but also for its $3^k - 1$ immediate neighbours.)

At the end of this process we determine which *(model curve,shift)* pairs got the most votes, and for every such pair determine approximate starting and endpoints of match between the signature sequence of the observed curve and the signature sequence of the model curve under the appropriate shift. This is done by aligning the signature sequences for the candidate according to the appropriate shift and finding the longest consecutive matching subsequence (allowing minor mismatches). Given these subsequences we find the actual subcurves to which they correspond and then apply the robust Schwartz-Sharir matching algorithm (see [SS 85]), which is of complexity $O(nlogn)$ (n-number of sample points on the longer curve).

but reduces to complexity $O(n)$ in our case. The Schwartz-Sharir algorithm produces the rotation and translation parameters needed to match one curve to the other and also a L_2 goodness of fit score between the curves along the matching subcurve. Given this rotation and translation, we can align both curves and re-determine their longest matching subcurve in true 3-D coordinates. Combining the evidence on the length of fit and score we decide which of the candidate pairs *(model curve, shift)* obtained from the geometric hashing process represents the correct match. As mentioned above, signature sequence hashing is only used to filter out inappropriate candidates, and the few remaining 'strong' candidates are distinguished using the 3-D subcurve matching algorithm due to Schwartz and Sharir.

3.3 Summary of Algorithm Steps

To summarize our algorithm :

A We represent model curves by sequences of characteristic signatures which are local translationally and rotationally invariant, and have satisfactory separation properties. We hash these signatures into a table which stores all pairs *(curve number, sample point number)* for every signature.

B Given an observed curve we compute its signature sequence and find the pairs *(curve number, relative shift)* which obtained the most coincidence votes. For each such candidate pair we align the signature sequences according to the appropriate shift and find the longest consecutive matching subsequence (allowing minor mismatches). Candidates which have a long enough consecutive matching subsequences are passed to the next step.

C For each subsequence passed from *Step B*, we go back to the original curves and match the two subcurves which correspond to this subsequences using the Schwartz-Sharir subcurve matching algorithm, thus determining the desired translation and rotation of one curve with respect to the other, and the goodness of the proposed fit. We discard candidate matches with 'poor' fit.

D For each candidate match from the previous step, we rotate and translate the curves as the match specifies, and redetermine the longest matching subcurves of the two curves, given this rotation and translation. This subcurve is found by simply checking the (x,y,z) coordinates of corresponding points on the curves and demanding that the distance between the points should be less than a certain threshold value ϵ. This final check works with points on the curves themselves, and not with the (less accurate) signature sequence values at these points; hence it is quite robust.

E The result giving the longest matching subcurve (allowing minor mismatches) with a good L_2 fit is chosen as the final solution.

3.4 Complexity Analysis and the Weighted Signature Approach

The algorithm that we have described is on the average linear in the number of sample points on the observed curve, and the computational cost is relatively independent of the number of points on the model curves in the data-base. (As noted previously in matching of two curves we have only one model curve, hence the hash table

records only the indices of the sample points where a given signature appears.)

This method can be improved by introducing *weighted* signatures. Specifically, since for typical curves not all the signatures will have an equal probability of occurrence, it seems undesirable to give an equal weight to every 'hit', but to weight coincidence of 'rare' signatures. The actual probability of occurrence of each individual signature can be estimated by the number of its occurrences in the database, which naturally can serve as a statistical sample for this data. Such a weighted signature approach can also improve the efficiency of our algorithm by making it linear instead of linear on the average. This can be done by assigning zero weight to very frequent (more than a certain predefined constant threshold K) signatures thereby eliminating the need to process hash-table entries representing many candidates; such entries require much computer time but contribute only a small amount of information to the matching process.

A major potential advantage of the algorithm presented is its high inherent parallelism. Since all the signatures can be processed independently, parallel implementation of both the *Preprocessing* and *Matching* parts of the algorithm is straightforward; moreover, it should be quite easy to build a special device for this implementing it at very high speed.

4 Experimental Results

A series of experiments was carried out to test and evaluate the performance of the matching algorithm with real 3-D data. Pieces of a plastic ball were chosen as experimental objects. Piecewise linear approximations to the boundary curves of these pieces were extracted from the range data obtained with a *Technical Arts Corporation* laser rangefinder. The 3-D curves of the different pieces were matched against each other in order to find how these pieces should be assembled together. Figure 2 shows a plastic ball, and the same ball after it was cut into 11 pieces and then reassembled. (Since at this point we are only interested in evaluating the matching algorithm, we did not actually attempt to reconstruct the ball. The full reconstruction of the ball would require global consistency between different matches, since the correct match between curves may not necessarily be the longest possible close match. See [KLSW 86] for a detailed discussion of a 'jigsaw puzzle assembly' problem in two dimensions).

4.1 Data Acquisition

To match 3-D objects by our method 3-D coordinates of points lying on curves along the object are needed; hence we acquire range information. Range data can be gathered in many ways (see [BJ 85] for a survey of range gathering techniques). In our application we used a plane of light, laser based range sensor [WS 83]. (See Figure 3 for a perspective view of the range image of one of the pieces.) The range data obtained are fairly accurate, so no initial smoothing or

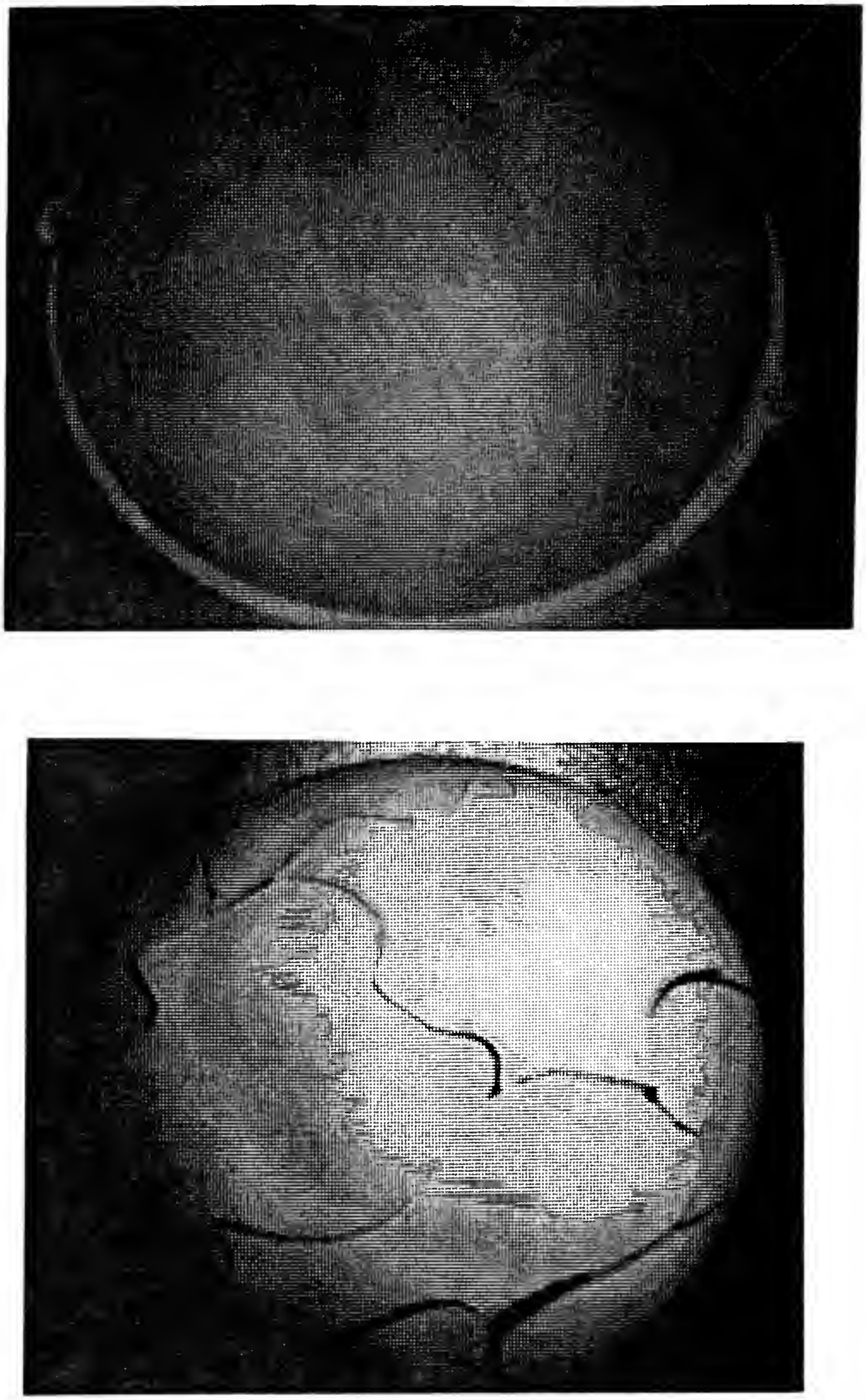


Figure 2: a) a plastic ball. b) the separate pieces of the ball put together.

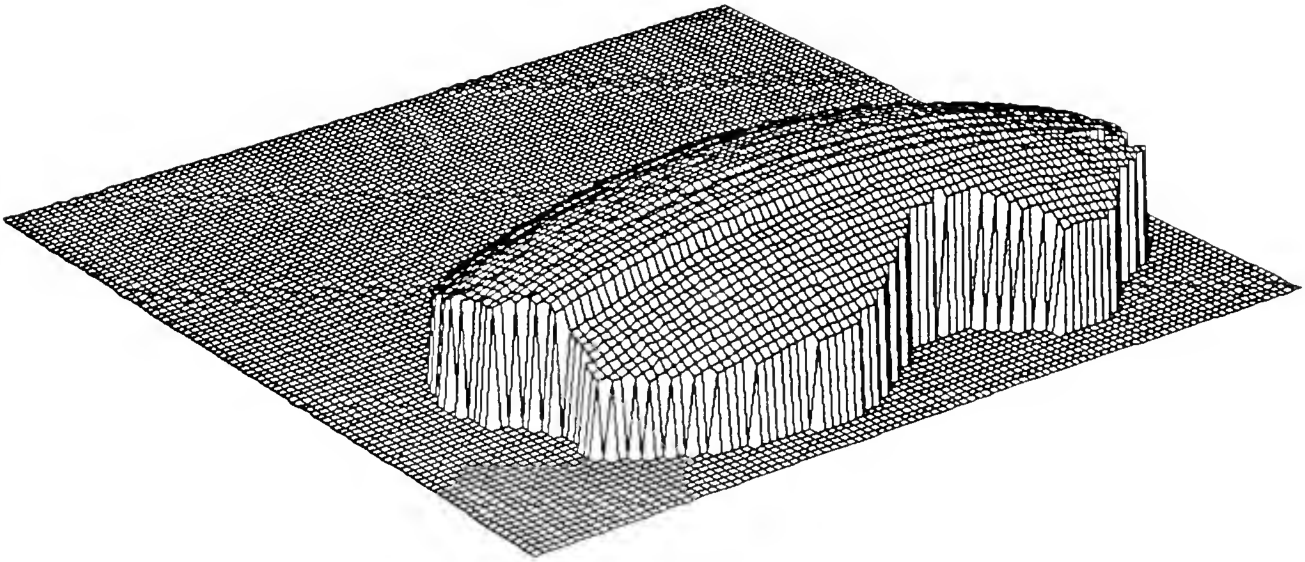


Figure 3: range image of a piece

averaging of the raw range data before edge extraction is necessary. Boundary edges of the plastic pieces used are well defined by large discontinuities in the range data, allowing a simple gradient based edge detection operator to extract the boundary edges of the pieces. The next step is to walk around each boundary in the range image and connect all edge fragments into one curve, while registering the 3-D coordinates of the points lying along the curve.

4.2 Processing of 3-D curves

Figure 4 shows the 3-D curve extracted from piece of Figure 3. Although no smoothing or averaging were applied to the data the Z-coordinate of this 3-D curve is smooth as we see from Figure 5. Nevertheless the XY projection of this curve is not perfectly smooth because of

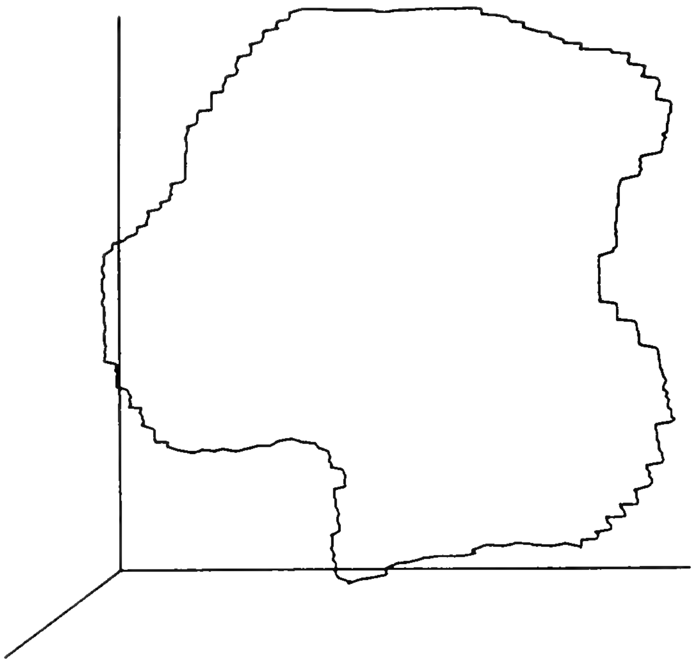


Figure 4: the boundary curve of a piece

discretization errors. A pixel in the camera has a finite extent, and so a pixel on the edge of an object would receive light not only from the portion of the pixel that is on the object, but also from the portion of the pixel that covers the background. As a result, some pixels on the boundary of an object will be considered as part of the object, while other pixels will be considered as background, and so the the boundary of an object (even a smooth object) will be noisy. Our algorithm uses the arc length of a curve in an important way; this forces us to smooth before going on. However, since only the XY projection of the 3-D curve is noisy, we only want to smooth this 2-D projection of the curve. Hence we use the Z coordinates of the original 3-D curve and the smoothed 2-D curve to reconstruct a smoothed version of the 3-D curve. Two issues arise in the smoothing process: the length of the resulting smoothed curve should be insensitive to random noise

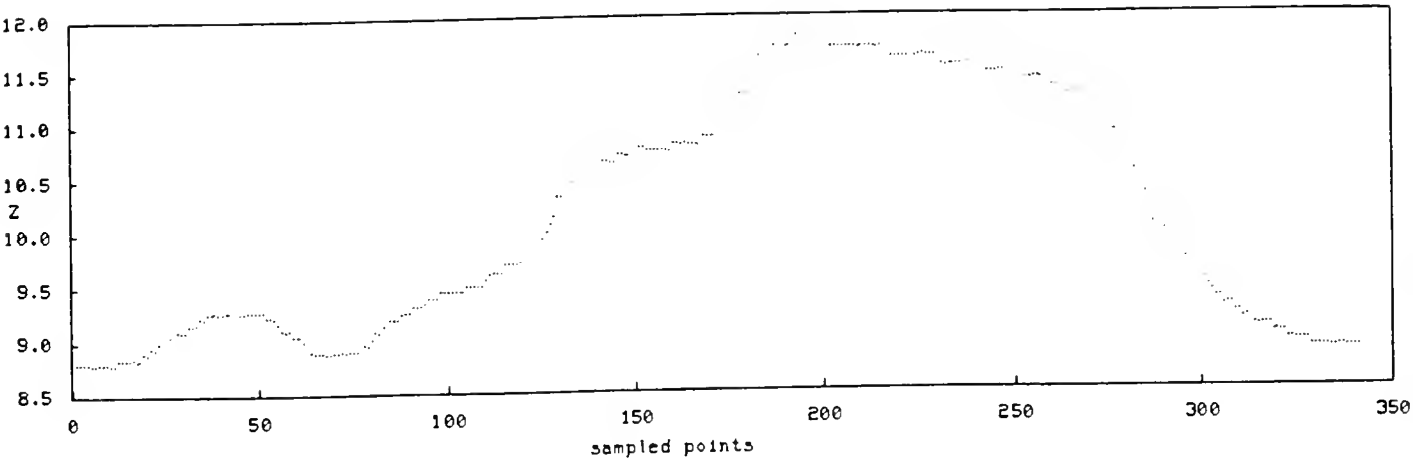


Figure 5: Z-coordinates of a 3-D curve ploted against the sample points

in the original curve (because of the parametrization by arc length), and should preserve local curvatures as much as possible (so as not to distort the local signature that we use in the algorithm).

2-D smoothing of a curve is achieved by surrounding the original curve by a 'belt' whose width is an estimated noise parameter ϵ. and then finding the shortest polygonal path in the resulting 'belt'. (A detailed description of this technique is found in [SS 85].) Figure 6 shows the raw 3-D curve of a piece, and three smoothed versions of the same curve, obtained with different error parameters ϵ.

4.3 Matching results

Figure 7 shows four different pieces of a plastic ball, and Figures 8, 9 and 10 show the results of matching individual pieces of the ball and finding the best subcurve match. Notice that the subcurves blend smoothly in the match, so that the boundaries of the subcurves to be matched cannot be distinguished by any sharply defined features.

The number of points used to represent the smoothed curves is about

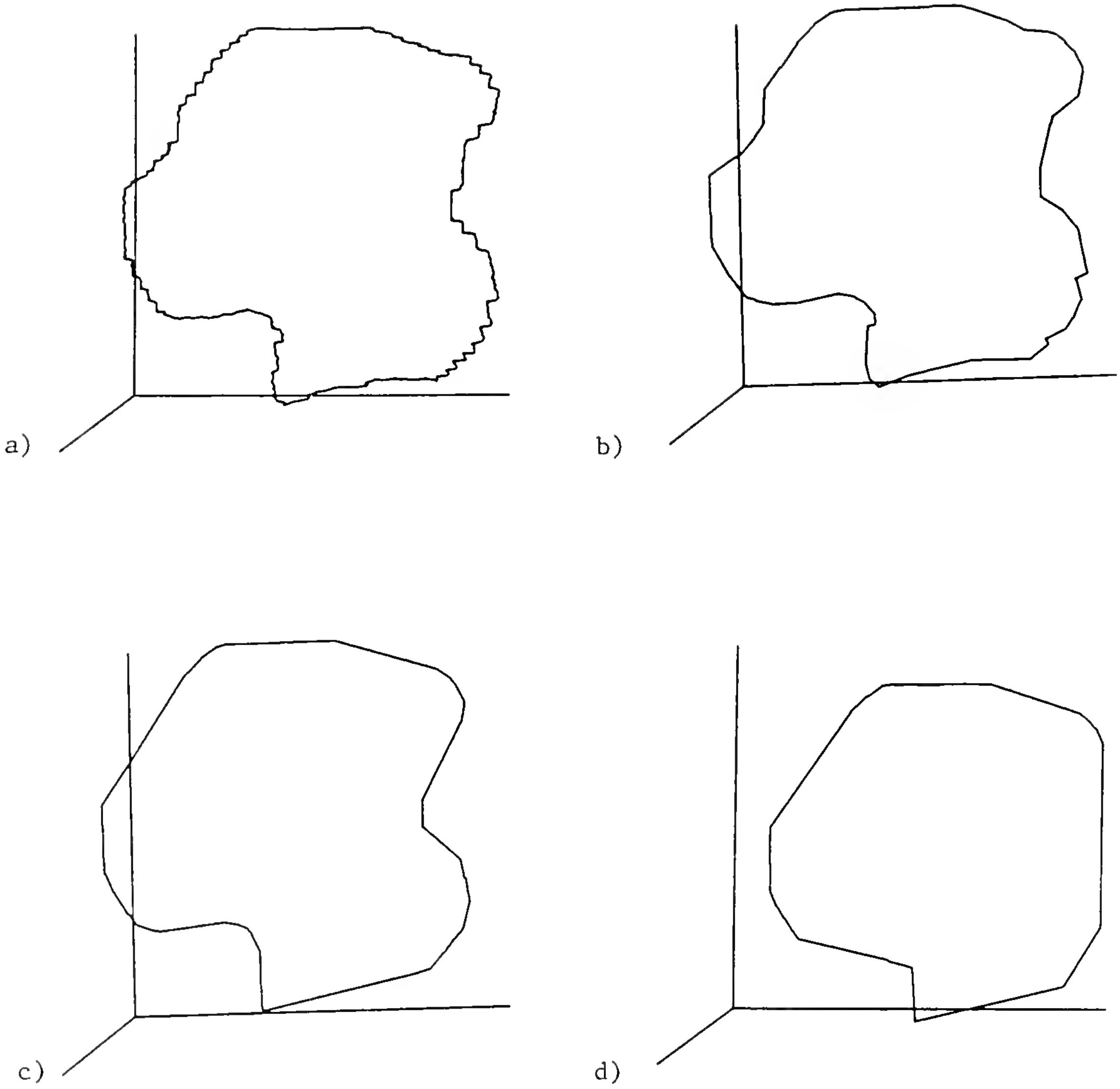


Figure 6: a) the original curve. b) the curve smoothed with epsilon .1 c) the curve smoothed with epsilon .2 d) the curve smoothed with epsilon .3

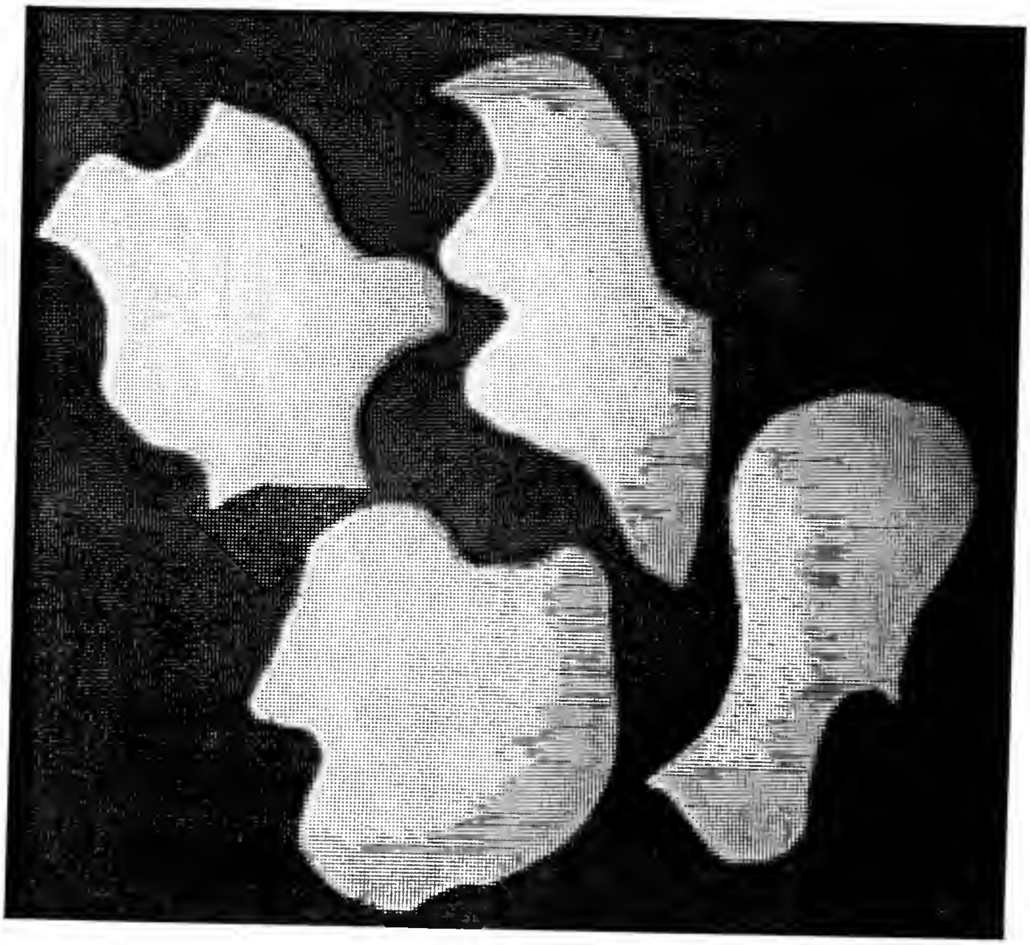
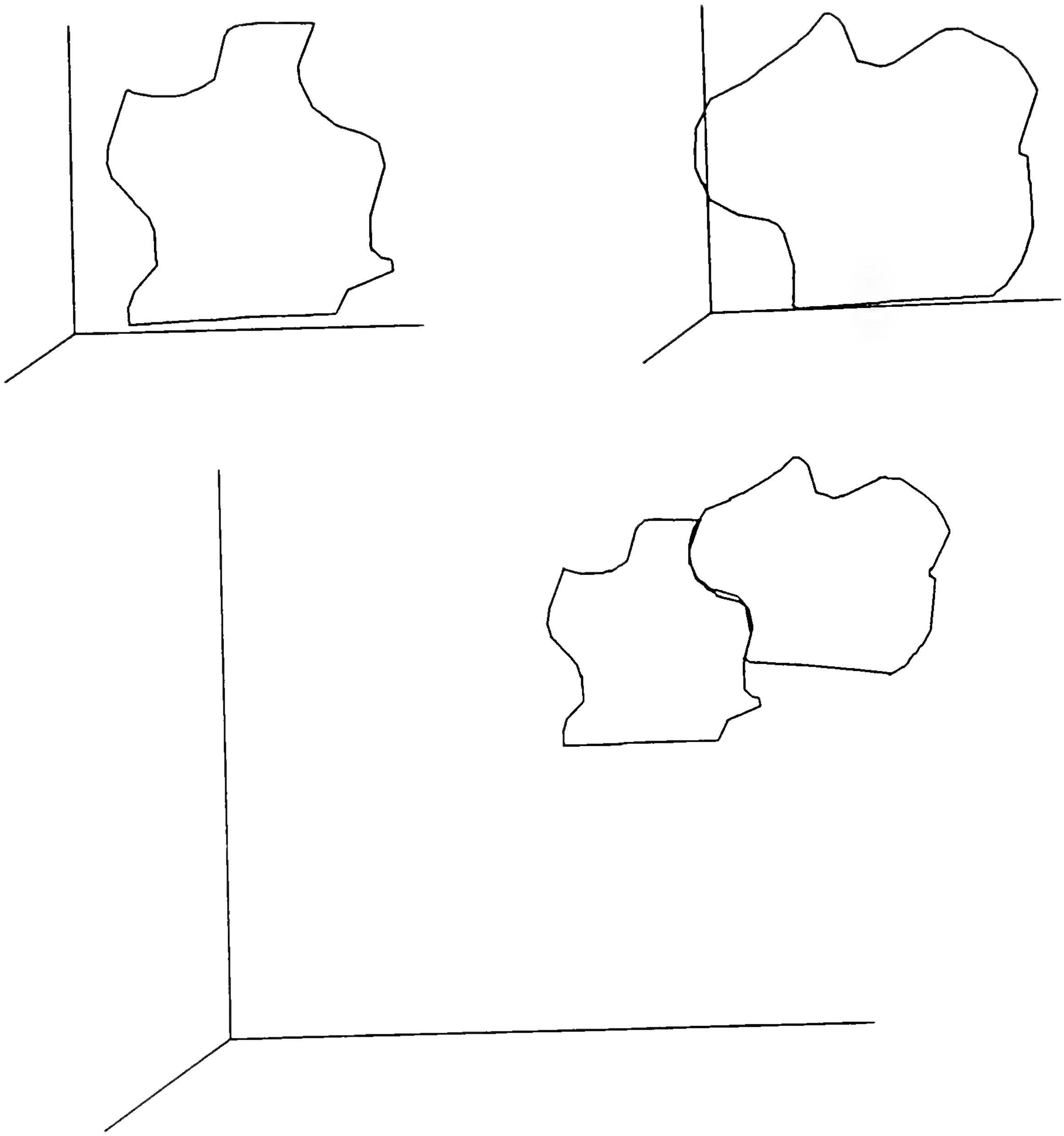


Figure 7: four pieces of the plastic ball



a)

b)

Figure 8: a) the boundary curves of two pieces. b) the result of matching the two boundary curves.

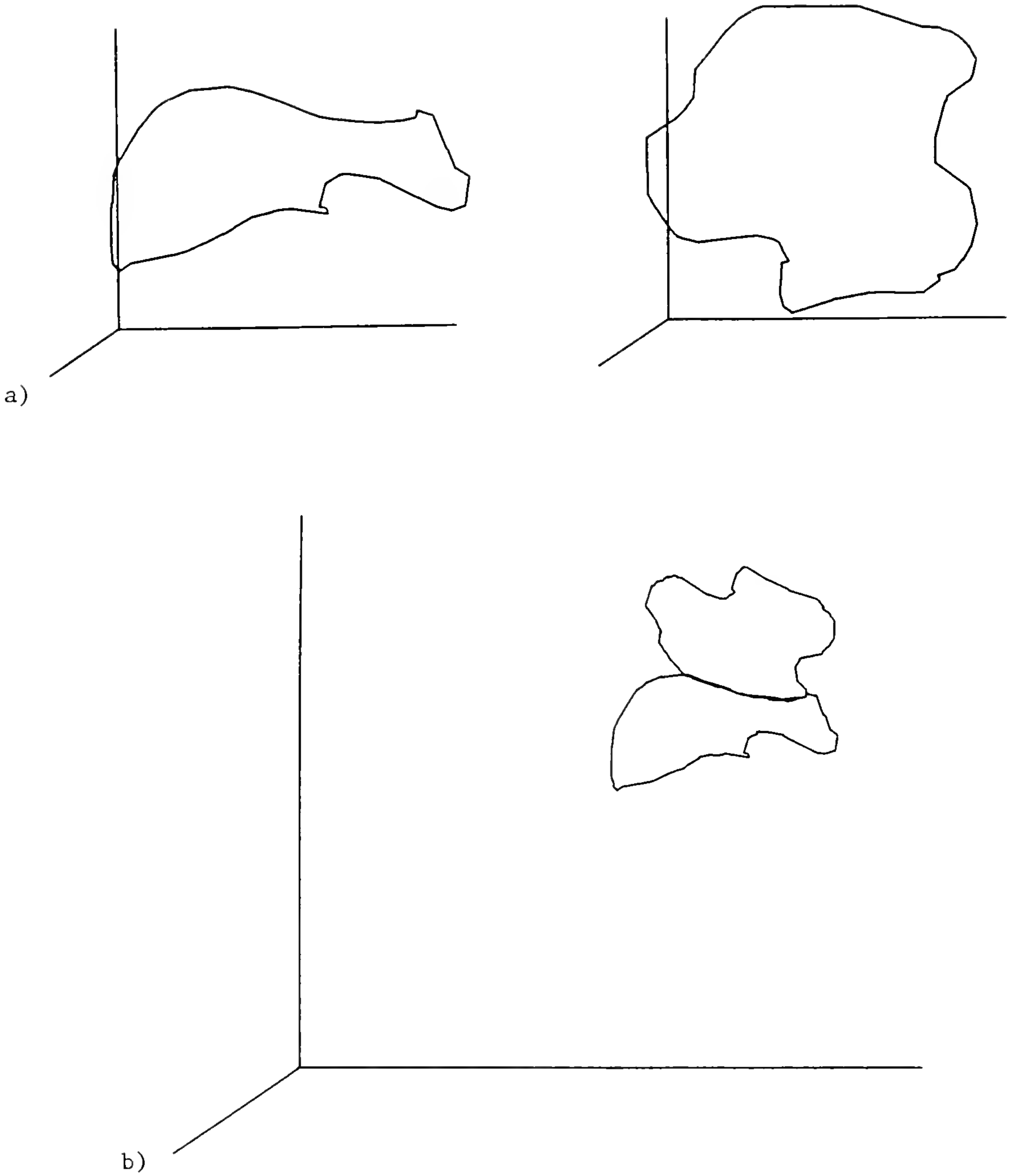
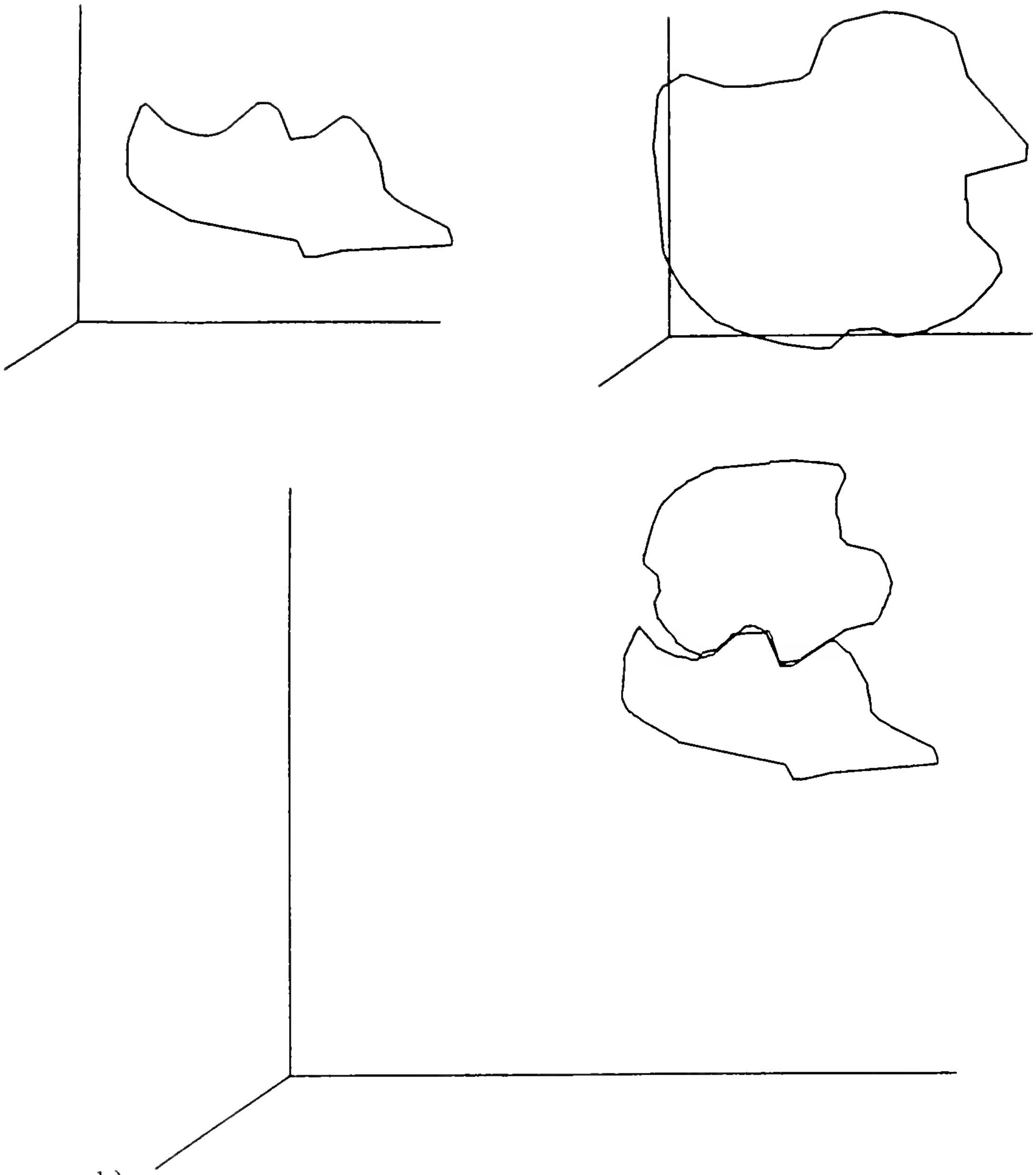


Figure 9: a) the boundary curves of two pieces. b) the result of matching the two boundary curves.



a)

b)

Figure 10: a) the boundary curves of two pieces. b) the result of matching the two boundary curves.

100, and since the matching algorithm is linear on the average, running time for a typical match was less than 1 second on a MicroVax.

5 Future Research

The matching algorithm presented in this article can serve as a key ingredient in a number of important robotic vision tasks. Applications and extensions with which we intend to deal are as follows:

1. Extension of the partial curve matching technique to a global technique for reassembling broken objects.
2. Definition and extraction of significant '*characteristic*' curves on a 3-D object.
3. Systematic study of 3-D object models based on '*characteristic*' curves.
4. Construction of curve based models of 3-D object using views acquired from a small number of viewpoints. The matching algorithm can be applied here to assemble such models.

Bibliography

[B81] C.M.Brown, Some mathematical and Representational Aspects of Solid Modeling, *IEEE trans., PAMI-3,4*, July, 1981.

[BB82] D.H.Ballard and C.M.Brown, *Computer Vision*, Prentice Hall, 1982.

[BJ85] P.J.Besl and R.C.Jain, "Three-Dimensional Object Recognition", *ACM computing surveys*, March, 1985.

[BS86] C.M. Bastuscheck, and J.T. Schwartz, "Experimental Implementations of a Ratio Image Depth Sensor", *Techniques for 3-D machine perception*, A. Rosenfeld, Ed. North Holland, 1986.

[$BSSS$86] C.M. Bastuscheck, E. Schonberg. J.T. Schwartz, and M. Sharir, "Object Recognition by 3-Dimensional Curve Matching", *Int. J. of Intelligent systems, Vol.1, No.2*, 1986.

[CD86] R.T.Chin and C.R.Dyer, "Model-Based Recognition in Robot Vision", *ACM computing surveys, Vol.18, No.1*, March, 1986.

[D76] M.P. Do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice Hall, 1976.

[HW] J.W. Hong, and H.Wolfson, "On Footprints", *in preparation.*

[$K86$] E.Kishon, "Simultaneous Matching of Curves in Three Dimensions", *Robotics Report No. 91, Computer Science Div., Courant Inst. of Math., NYU*, Nov. 1986.

[$KLSW86$] A.Kalvin, Y.Lamdan, E.Schonberg, H.Wolfson, "Solving Jigsaw Puzzles Using Computer Vision", *Robotics Report No. 65. Computer Science Div., Courant Inst. of Math., NYU*. Feb. 1986.

[$KSSS87$] A. Kalvin, E. Schonberg, J.T. Schwartz, and M.Sharir, "Two Dimensional, Model Based, Boundary Matching Using Footprints", *Robotics Research, Vol.5. No.4*, 1987.

[$S69$] J.J. Stoker, *Differential Geometry* , Wiley-Interscience, 1969.

[$SS85$] J.T. Schwartz, and M. Sharir, "Identification ot Partially Obscured Objects in Two or Three Dimensions by Matching of Noisy Characteristic Curves", *Robotics Report No. 46. Computer Science Div., Courant Inst. of Math., NYU*, 1985.

[$W86$] H. Wolfson, "On Curve Matching", *Robotics Report No. 86, Computer Science Div., Courant Inst. of Math., NYU*, 1986.

[WS83] "White scanner 100A user's manual", Technical Arts Corp, Seattle.

NYU COMPSCI TR-283 c.1
Kishon, Eyal
3-D curve matching.